SRC TR 90-25

A Structured Fixed-Rate Vector Quantizer Derived from Variable-Length Encoded Scalar Quantizers

By

R. Laroia and N. Farvardin

A Structured Fixed-Rate Vector Quantizer Derived from Variable-Length Encoded Scalar Quantizers †

by

R. Laroia and N. Farvardin
Electrical Engineering Department
Institute for Advanced Computer Studies
and Systems Research Center
University of Maryland
College Park, Maryland 20742

Abstract

The well-known error propagation problem inherent in any variable-length coding operation limits the usefulness of variable-length encoded entropy-constrained scalar quantizers when the quantizer outputs are to be transmitted over a noisy channel. In the absence of channel noise, however, these quantizers are known to perform better than error-minimizing fixed-rate Lloyd-Max quantizers for a wide class of memoryless sources. Motivated by this observation, in this paper we develop a fixed-rate vector quantization scheme which achieves performance close to that of optimum entropy-constrained scalar quantizers; due to the fixed-rate nature of the encoder, channel error propagation is not an issue any more. An algorithm for the design of this scheme is described and procedures for codebook search and codevector encoding are developed. We show that codebooks significantly larger than those in conventional vector quantizers can be designed. Numerical results demonstrating the efficacy of this scheme along with comparisons against Lloyd-Max quantizers and optimal entropy-constrained quantizers are rendered.

† This work was supported in part by National Science Foundation grants NSFD MIP-86-57311 and NSFD CDR-85-00108, and in part by a grant from General Electric.

I. Introduction

Optimal entropy-constrained scalar quantizers (ECSQ's) are known to perform close to the rate-distortion bound for a large class of memoryless sources [1], [2]. To implement such quantizers, however, some type of *variable-length* noiseless coding is needed. Due to the sequential nature of decoding of such codes, channel errors could lead to a loss of synchronization resulting in propagation of error and hence large cumulative reconstruction errors. Therefore, ECSQ's are of limited usefulness for most practical applications that involve transmission over noisy channels; usually some type of packetization to limit the error propagation to within a packet is used.

Optimal minimum-distortion (Lloyd-Max) quantizers (LMQ's) utilizing *fixed-length* codewords, on the other hand, are immune to error propagation and related problems but in the absence of channel noise perform worse than ECSQ's [3],[4]. The gap between the performance of optimal ECSQ's and LMQ's can be large for certain sources [1]. To bridge this gap while maintaining fixed-length codewords one could use a vector quantizer (VQ). Optimal *fixed-rate* VQ's of sufficiently large dimension can perform arbitrarily close to the rate-distortion bound. Such quantizers however have two drawbacks: (i) the encoding complexity (which grows exponentially with the rate and block-length) and (ii) the large amount of memory required to store the codebook. The problem of encoding complexity can be alleviated, at the cost of a small performance degradation, by designing suboptimal VQ's such as tree-searched VQ's [5]. But, this only adds to the storage problem. In any practical situation, the complexity and memory problems place a constraint on the maximum codebook size and hence the performance of the VQ.

The permutation encoder (PE) is a *fixed-length* encoder which can be thought of as a VQ with a special structure (the codebook consists of codevectors that are permutations of one another) [6]. The structure of this codebook obviates the need for storing the codevectors and also results in a simple encoding algorithm. It is proved in [6] that the average distortion of the PE is bounded below by that of the optimal ECSQ at the same rate and in the limit of infinite block-length, the optimal PE becomes equivalent to the optimal ECSQ. The block-length of the PE required to achieve performance better than LMQ is very large resulting in large encoding delays. In addition, while the channel errors do not propagate beyond block boundaries, they do affect all samples within a block; therefore, very large block-lengths offset some of the advantages of fixed-length coding

over noisy channels.

This paper describes a *fixed-length structured vector quantizer* (SVQ) based on variable-length encoded scalar quantizers. The fidelity criterion used here is the squared-error distortion. The basic idea can be described as follows. Consider an ECSQ the outputs of which are encoded by variable-length codewords of length given by the negative logarithm of the probability of the corresponding quantization level. If such a variable-length encoded ECSQ (VL-ECSQ) is used to quantize a block of samples from a memoryless source, the total length of the output reflects its probability (smaller length means higher probability). When the block-length is large, with a very high probability the length of the output is close to the typical length $\equiv$(*block-length* $\times$ *entropy*). This suggests that if (for large block-length) the output length of a VL-ECSQ is constrained to lie in a small neighborhood of the typical length, then with high probability the quantizer outputs can be encoded without any distortion. If all the possible VL-ECSQ outputs that satisfy the length constraint are counted and encoded with fixed-length codewords, the system will perform close to the ECSQ. Although the performance of this system approaches that of the ECSQ as the block-length goes to infinity, for any fixed block-length it is preferable to encode all the VL-ECSQ outputs of lengths equal to or less than the typical length. This is because the smaller length outputs are more probable. This idea provides the motivation for the present work, the actual scheme as formally described in Section II is slightly different and uses an optimized scalar quantizer in place of an ECSQ.

A procedure for the design of the encoder parameters is described in Section III. Section IV addresses the implementation issues which include codebook search and codevector encoding. Comparisons between the SVQ and some other schemes (mentioned above) are made in Section V while the storage and the computational requirements of the SVQ are tabulated in Section VI. Specific numerical results demonstrating the efficacy of this scheme for a variety of sources, encoding rates and block-lengths are presented in Section VII. Finally, a summary and conclusions are presented in Section VIII.

II. The Structured Vector Quantizer

For the rest of this paper we will assume that the source to be encoded is a stationary memoryless source. In the structured vector quantization scheme proposed here, the codebook structure is derived from a variable-length encoded scalar quantizer. If each

component of an m-vector is separately quantized using an n-level scalar quantizer, the set of possible output vectors form an m-dimensional grid of n^m points which is contained inside an m-cube. For an m-dimensional SVQ each codevector corresponds to a point on this grid. However, not every grid-point has a codevector associated with it. To determine the grid-points to be included in the codebook we first assign some lengths ℓ_i, $i \in J_n$, $(J_n \equiv \{1, 2, \ldots, n\})$ to the quantization levels q_i of the underlying scalar quantizer. These lengths are not required to be integers, but are rounded-off to the nearest multiple of some fraction $1/b$ of a bit. Next, we determine the total length associated with each grid-point as the sum of the lengths of its m components. If the total length of a point is less than a threshold L, we include the point in the codebook. The threshold is determined so that the codebook contains 2^{mr} codevectors where r is the desired average transmission rate in bits/sample. With this structure of the codebook it is possible to quantize the input vector to the nearest codevector without explicitly storing the codebook (Section IV). An algorithm for determining the set of scalar quantizer levels $\mathcal{Q} \equiv \{q_1, q_2, \ldots, q_n\}$, the set of lengths $\mathcal{L} \equiv \{\ell_1, \ell_2, \ldots, \ell_n\}$ and the threshold L is presented next.

III. Design of the SVQ

The SVQ is completely described by $\mathcal{Q}$, $\mathcal{L}$ and L. Designing the SVQ hence corresponds to the determination of these quantities. For a given source and an average transmission rate r, the SVQ can be designed using a two step iterative process. The first step (Step A) determines the quantization levels $\mathcal{Q}$ given $\mathcal{L}$ and L. The second step (Step B) determines the lengths $\mathcal{L}$ and threshold L given $\mathcal{Q}$. If both of these steps are performed optimally (with respect to some distortion measure), then each iteration reduces distortion and since the distortion is lower bounded by zero, the solution converges to a locally optimal design. Unfortunately, the optimal solution for $\mathcal{L}$ and L given $\mathcal{Q}$ in Step B is not known for the squared-error distortion measure. These quantities are hence determined suboptimally resulting in a suboptimal design of the SVQ. Step A and Step B of the design algorithm are described below.

A. Step A

The problem of determining $\mathcal{Q}$ optimally given $\mathcal{L}$, L and a source probability density

function (*pdf*) is itself solved iteratively [1] using a method similar to the optimal design of conventional VQ's [7]. A long training set of N source vectors is generated and this set is used to characterize the *pdf* of the source. Let $\mathcal{X} = \{\mathbf{x}_j; j \in J_N\}$ denote the training set of m-vectors $\mathbf{x}_j = (x_{1j}, x_{2j}, \ldots, x_{mj})$. Given the set of quantization levels $\mathcal{Q}^k$ (where k is the iteration index), quantize each vector in $\mathcal{X}$ using the SVQ defined by $\mathcal{Q}^k$, $\mathcal{L}$ and L. Let $\mathcal{Y}^k = \{\mathbf{y}_j^k; j \in J_N\}$ represent the set of quantized vectors where $\mathbf{y}_j^k = (y_{1j}^k, y_{2j}^k, \ldots, y_{mj}^k)$ and $y_{ij}^k \in \mathcal{Q}^k$, $i \in J_m$, $j \in J_N$. The mean squared-error (MSE) D for this quantization operation can be expressed as,

$$\begin{aligned} D &= \frac{1}{N}\sum_{j=1}^{N} ||\mathbf{x}_j - \mathbf{y}_j^k||^2 = \sum_{j=1}^{N}\sum_{i=1}^{m}(x_{ij} - y_{ij}^k)^2 \\ &= \frac{1}{N}\sum_{l=1}^{n} \sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} (x_{ij} - q_l^k)^2. \end{aligned} \tag{1}$$

The new $\mathcal{Q}$ given by $\mathcal{Q}^{k+1} = \{q_1^{k+1}, q_2^{k+1}, \ldots, q_n^{k+1}\}$ where

$$q_l^{k+1} = \left[\sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} 1\right]^{-1} \left[\sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} x_{ij}\right], \quad l \in J_n, \tag{2}$$

minimizes D and hences reduces distortion over $\mathcal{Q}^k$.

Since each step of this iterative process reduces the MSE, $\mathcal{Q}^k$ converges to a locally optimal solution $\mathcal{Q}_{opt}(\mathcal{L}, L)$.

B. Step B

As mentioned before, the optimal solution for $\mathcal{L}$ and L given $\mathcal{Q}$ is not known. The following suboptimal solution is proposed instead. Since the m-dimensional grid (Section II) contains n^m points out of which we want to choose only 2^{mr} points as codevectors, we

[1] For implementing the algorithm to design SVQ's it is not necessary to keep performing the iterations within Step A until the solution for $\mathcal{Q}$ converges. In fact only one such iteration needs to be performed each time Step A is visited.

choose the 2^{mr} most probable ones. Clearly this choice does not necessarily minimize the MSE, but it ensures that with the highest probability the input vector will be quantized to the nearest point on the grid. The usefulness of this suboptimal solution is supported by the results given in Section VII.

Let p_i, $i \in J_n$ be the probability that the input sample is quantized to level q_i if scalar quantization to the nearest level is performed. These probabilities can either be calculated from the source *pdf* or the set of training vectors. The probability $P(\mathbf{v})$ of an arbitrary grid-point $\mathbf{v} = (v_1, v_2, \ldots, v_m)$ with $v_i \in \mathcal{Q}$, $i \in J_m$, is then given by (*iid* input),

$$P(\mathbf{v}) = \prod_{l=1}^{m} p_{f(v_l)} \quad = 2^{-\sum_{l=1}^{m} \log(1/p_{f(v_l)})} , \tag{3}$$

where the index function $f : \mathcal{Q} \rightarrow J_n$ is defined as,

$$f(q_i) = i, \quad i \in J_n . \tag{4}$$

We define the length $\ell_i = \log 1/p_i$, $i \in J_n$, resulting in

$$P(\mathbf{v}) = 2^{-\sum_{l=1}^{m} \ell_{f(v_l)}}. \tag{5}$$

This shows that the higher probability grid-points are the ones with smaller total lengths. Hence choosing the 2^{mr} most probable grid-points corresponds to choosing 2^{mr} points with the smallest total lengths.

The suboptimal threshold L is given by the maximum value of the threshold for which there are at most 2^{mr} vectors in the codebook. An algorithm to do this follows.

C. Determination of the threshold L

The threshold L is obtained by counting the grid-points (starting with the ones that have the smallest total length) until we have 2^{mr} points and taking the largest total length in this collection.

Let N_i^j represent the number of distinct i-vectors $(z_1, z_2, \ldots, z_i)$ with $z_1, z_2, \ldots, z_i \in \mathcal{Q}$ such that their total length $\ell_{f(z_1)} + \ell_{f(z_2)} + \ldots + \ell_{f(z_i)} = j$. Then N_i^j satisfies the following recursive equation $\forall\, i \in J_m$:

$$N_i^j = \sum_{k=1}^{n} N_{i-1}^{j-\ell_k} , \tag{6}$$

where $N_i^j = 0$ for $j < 0$ and $N_0^0 = 1$. The N_i^j's can hence be determined using a dynamic programming type of algorithm. The total number of grid-points C_m^j with total length less than or equal to j is now given as,

$$C_m^j = \sum_{k=1}^{j} N_m^k \ . \tag{7}$$

The threshold L can therefore be evaluated as,

$$L = \max\{j : \ C_m^j \leq 2^{mr}\} \ . \tag{8}$$

This choice of L guarantees that there will be at most 2^{mr} vectors in the codebook and hence each codevector can be encoded by an mr-bit binary codeword.

IV. Implementation of the SVQ

The encoding operation essentially involves two steps, the codebook search and the encoding of the codevectors. Fast algorithms for both of these steps are now presented.

A. Codebook search

The codebook search can be performed by a dynamic programming algorithm similar to the Viterbi algorithm for convolutional coding [8]. The aim here is to find a grid-point of total length less than or equal to L that is closest (in the sense of minimum distortion) to the given input vector. As mentioned before, we assume that the lengths ℓ_i, $i \in J_n$ and the threshold L are rounded-off to the nearest multiple of a fraction $1/b$ of a bit. Using this fraction as a unit of length measurement, we can represent all lengths as positive integers. Without loss of generality we assume that ℓ_i, $i \in J_n$ and L are positive integers.

Let D_i^j be the minimum squared-error distortion that results when the first i components of the input m-vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ are quantized to any of the i-vectors $\mathbf{z}_i = (z_1, z_2, \ldots, z_i)$ with $z_1, z_2, \ldots, z_i \in \mathcal{Q}$ such that the total length of $\mathbf{z}_i$ is j, i.e., $\ell_{f(z_1)} + \ell_{f(z_2)} + \ldots + \ell_{f(z_i)} = j$. Also, let $\mathbf{z}_i^j$ represent the i-vector that results in the minimum distortion D_i^j. The distortion D_{i+1}^j is then recursively given as,

$$D_{i+1}^j = \min_{k \in J_n} \left[D_i^{j-\ell_k} + (x_{i+1} - q_k)^2 \right], \ i = 0, 1, \ldots, \ m-1, \tag{9}$$

where $D_i^j = \infty$ for $j \leq 0$ or $i \leq 0$ but $D_0^0 = 0$.

If the above is minimized for $k = k'$, then the corresponding minimum distortion vector $\mathbf{z}_{i+1}^{j}$ is given by the recursive equation,

$$\mathbf{z}_{i+1}^{j} = (\mathbf{z}_{i}^{j-\ell_{k'}}, q_{k'}) \; . \tag{10}$$

Using the dynamic programming algorithm to solve these equations we can determine D_m^j, $\forall j \in J_L$. The minimum distortion $D_{\min}$ is then given as,

$$D_{\min} = \min_{j \in J_L} \; D_m^j \; . \tag{11}$$

If $j = j'$ minimizes this distortion then $\mathbf{z}_m^{j'}$ gives the desired codevector.

B. Encoding of codevectors

There are 2^{mr} codevectors in the codebook $\mathcal{Z}$ of the SVQ and the encoder is a mapping that assigns mr-bit binary numbers to the codevectors in a one-to-one manner. The following algorithm implements one such mapping.

To every codevector $\mathbf{z} = (z_1, z_2, \ldots, \; z_m) \in \mathcal{Z}$ is assigned an m-digit base-n number $\mathcal{N}(\mathbf{z})$ given as,

$$\begin{aligned} \mathcal{N}(\mathbf{z}) &= (f(z_1) - 1, \; f(z_2) - 1, \; \ldots, \; f(z_m) - 1) \\ &= \sum_{l=1}^{m} n^{m-l}(f(z_l) - 1). \end{aligned} \tag{12}$$

Clearly, $\mathcal{N}(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_2) \Leftrightarrow \mathbf{z}_1 = \mathbf{z}_2$. The encoder function $E: \; \mathcal{Z} \rightarrow J_{2^{mr}}$ is defined as,

$$E(\mathbf{z}) = \sum_{\mathbf{w} \in \mathcal{Z}} X_{\mathbf{z},\mathbf{w}} \tag{13.a}$$

where

$$X_{\mathbf{z},\mathbf{w}} = \begin{cases} 0 & \text{if } \mathcal{N}(\mathbf{w}) \geq \mathcal{N}(\mathbf{z}) \\ 1 & \text{if } \mathcal{N}(\mathbf{w}) < \mathcal{N}(\mathbf{z}). \end{cases} \tag{13.b}$$

In other words, $E(\mathbf{z})$ is the number of codevectors in $\mathcal{Z}$ that are "smaller" than $\mathbf{z}$ (smaller in the sense $\mathcal{N}(\mathbf{w}) < \mathcal{N}(\mathbf{z})$). We can also write $E(\mathbf{z})$ as,

$$E(\mathbf{z}) = \sum_{k=1}^{m} E_k \; , \tag{14}$$

where E_k is the number of codevectors $\mathbf{w}$ such that $\mathcal{N}(\mathbf{w})$ and $\mathcal{N}(\mathbf{z})$ written as base-n numbers are equal in their $k-1$ most significant digits while the k^{th} digit of $\mathcal{N}(\mathbf{w})$ is smaller than that of $\mathcal{N}(\mathbf{z})$.

Let C_i^j represent the number of distinct i-vectors $(z_1, z_2, \ldots, z_i)$ with $z_1, z_2, \ldots, z_i \in \mathcal{Q}$ such that $\ell_{f(z_1)} + \ell_{f(z_2)} + \ldots + \ell_{f(z_i)} \leq j$. Also, define

$$C_0^j = \begin{cases} 0 & \text{if } j < 0 \\ 1 & \text{if } j \geq 0 \,. \end{cases} \tag{15}$$

It can easily be shown that

$$E_k = \sum_{j=1}^{f(z_k)-1} C_{m-k}^{L-L_{k-1}-\ell_j}, \quad k \in J_m \,, \tag{16}$$

where $L_0 = 0$ and $L_i = \sum_{j=1}^{i} \ell_{f(z_j)}$, $i \in J_m$.

Now (14) and (16) give,

$$E(\mathbf{z}) = \sum_{k=1}^{m} \sum_{j=1}^{f(z_k)-1} C_{m-k}^{L-L_{k-1}-\ell_j} \,. \tag{17}$$

Note that C_i^j can be evaluated by expressing it in terms of N_i^j (Section III) as,

$$C_i^j = \sum_{k=1}^{j} N_i^k, \quad \forall i \in J_m \text{ and } j > 0 \,. \tag{18}$$

For a fast implementation of this algorithm the C_i^j's can be evaluated once and stored in the memory.

V. Comparison with other schemes

In this section the performance of the SVQ is compared, in a qualitative manner, with the LMQ, the ECSQ and the permutation encoder. An SVQ for which Step B in the design algorithm of Section III is carried out optimally is referred to as an optimal SVQ.

An LMQ is a special case of the optimal SVQ when the block-length is one. Since it is reasonable to assume that the performance of the SVQ's improves with block-length,

the optimal SVQ always performs better than the LMQ. Our numerical results (Section VII) indicate that this conclusion also holds for the suboptimal SVQ of Section III.

It is difficult to analytically compare the performance of a fixed block-length SVQ to an ECSQ. But in the limit of infinite block-length it can be shown that the performance of an SVQ derived from an ECSQ (by using the quantization levels of the ECSQ and assigning to each level a length given by the negative logarithm of its probability) approaches the performance of the ECSQ. This suggests that the asymptotic performance of the optimal SVQ for large block-lengths is at least as good as that of the optimal ECSQ. This may not be true for the suboptimal SVQ.

A permutation encoder based on an ECSQ becomes equivalent to the ECSQ in the limit of infinite block-length. For relatively small block-lengths the permutation encoders perform poorly [6] and, as supported by the results in Section VII, the suboptimal SVQ's can be expected to perform better.

VI. Complexity issues

Table 1 gives the complexity of the codebook search and codevector encoding algorithms in terms of their computational and storage requirements. The computational complexity is measured by the number of equivalent 32 bit floating point operations (additions and comparisons) required per source sample. Based on a large number of SVQ's designed, the value of the threshold L is taken to be approximately $1.5mrb$ (where $1/b$ is the unit of length measurement). Since the number of quantizer levels n is expected to grow exponentially in r, the computational complexity of the SVQ grows linearly in m and exponentially in r while the memory requirement grows as the cube of m and exponentially in r. For large values of r a slightly modified implementation of the codebook search and the codevector encoding algorithms changes the exponential dependence of computational complexity on r to linear dependence [9]. With the present VLSI technology the complexity of an SVQ is quite affordable even for relatively fine quantization and a large block-length.

VII. Numerical results and comparisons

The performance results for SVQ's designed using the algorithm of Section III are reported in this section.

A. Convergence of the suboptimal design

Although the suboptimal algorithm for designing the SVQ's is not guaranteed to converge, for most SVQ's designed it was found that the solution actually converges. For a few cases it did not converge but the average distortion initially decreased and then oscillated in a narrow range. In such cases the SVQ with minimum distortion was selected.

B. The generalized Gaussian distribution

To analyze the performance of SVQ's, sources with a generalized Gaussian (GG) distribution were chosen because the GG distribution effectively models most unimodal symmetric distributions arising in practical situations. The *pdf* of the GG distribution is given by,

$$p(x) = \frac{\alpha\eta(\alpha,\beta)}{2\Gamma(1/\alpha)} \exp\left\{-[\eta(\alpha,\beta)|x|]^{\alpha}\right\}, \tag{19}$$

where $\eta(\alpha,\beta) = \beta^{-1}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2}$, $\alpha > 0$ is the exponential decay parameter and β^2 is the variance of the distribution.

The Laplacian and Gaussian distributions are special cases of the GG distribution for $\alpha = 1$ and 2, respectively. For large values of α the GG distribution approaches the uniform distribution while for small α it is a broad-tailed distribution.

C. Rate vs. distortion characteristics of the SVQ

Table 2 gives the signal-to-noise ratio (SNR) in *dB* of the SVQ's designed for GG sources with $\alpha = 0.5$, 1 and 2, for various rates and block-lengths. The corresponding SNR values of LMQ's and ECSQ's are provided for comparison. Fig. 1 gives plots of the average distortion as a function of the encoding rate for GG sources with $\alpha = 0.5$ and $\alpha = 2.0$ which show that the SVQ's effectively bridge the gap between LMQ's and ECSQ's while maintaining fixed-length outputs.

For all results in Table 2, length was quantized to multiples of a quarter of a bit ($b = 4$) and a training sequence of 50,000 source samples was used to characterize the source distribution. If the designed SVQ turned out to have a rate different from the desired rate[2], the SNR at the desired rate was obtained by interpolating the MSE.

[2] This could happen at low rates and small block lengths where the number of codevectors in the codebook could be significantly less than 2^{mr}.

D. Effect of the block-length m

In Fig. 2 the performance of the SVQ is plotted as a function of the block-length for GG sources with $\alpha = 0.5$ and $\alpha = 2.0$. These plots show that at low rates, a small block-length is adequate to perform close to the optimal ECSQ while for higher rates a larger block-length is required.

E. Effect of the number of quantization levels n

Fig. 3 shows the variation of the performance of an SVQ with the number of quantization levels. The two cases shown are: (i) GG source with $\alpha = 0.5$, $m = 16$ and $r = 2$ bits/sample, and (ii) GG source with $\alpha = 1$, $m = 32$ and $r = 3$ bits/sample. When $n = 2^r$, the SVQ reduces to m scalar LMQ's. For higher values of n the performance improves and then saturates. Broad-tailed distributions (smaller α) require a larger value of n to reach saturation.

F. Effect of length quantization

To reduce the complexity of implementation of the SVQ the lengths ℓ_i, $i \in J_n$ and threshold L are quantized to the nearest multiple of $1/b$ bits. The performance of the SVQ is plotted as a function of b in Fig. 4 for the GG sources with $\alpha = 0.5$, 1 and 2, $m = 16$ and $r = 2$ bits/sample. It can be seen that the performance does not depend significantly on b and does not necessarily improve as b increases. This should be expected because we use only a suboptimal algorithm for length assignment and hence small deviation in lengths (due to quantization) could improve or degrade performance.

VIII. Summary and conclusions

In this paper we have proposed a fixed-rate structured vector quantizer for encoding stationary memoryless sources; the the structure of the codebook is derived from a variable-length encoded scalar quantizer. A suboptimal algorithm for the design of the SVQ was presented along with fast algorithms for codebook search and encoding. The asymptotic performance of the optimal SVQ (for large block-lengths) was compared with some other quantization schemes.

Numerical results presented demonstrate that the SVQ's effectively bridge the gap between LMQ's and the optimal ECSQ's while maintaining fixed-rate outputs. The SVQ's perform close to the optimal ECSQ's at block lengths much smaller than those required by permutation encoders. Since the complexity of the SVQ's is affordable even for fine

quantization and relatively large block-lengths, the SVQ's can be used even when the vector quantizers are too expensive.

Better suboptimal (or possibly optimal) solutions for Step B of the design algorithm in Section III as well as extensions of SVQ to non-identically distributed sources and to sources with memory (Markov sources) are among interesting problems that deserve further investigation.

References

1. N. Farvardin and J.W. Modestino, "Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources," *IEEE Trans. Inform. Theory*, Vol. IT-30, pp. 485-497, May 1984.
2. J. Ziv, "On Universal Quantization," *IEEE Trans. Inform. Theory*, Vol. IT-31, pp. 344-347, May 1985.
3. S.P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Inform. Theory*, Vol. IT-28, pp. 129-137, March 1982.
4. J. Max, "Quantizing for Minimum Distortion," *IRE Trans. Inform. Theory*, Vol. IT-6, pp. 7-12, March 1960.
5. R. Gray and Y. Linde, "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources," *IEEE Trans. Commun.*, Vol. COM-30, pp. 381-389, Feb. 1982.
6. T. Berger, "Optimum Quantizers and Permutation Codes," *IEEE Trans. Inform. Theory*, Vol. IT-18, pp. 149-157, Nov. 1982.
7. Y. Linde, A. Buzo, R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, Vol. COM-28, pp. 84-95, Jan. 1980.
8. A.J. Viterbi and J.K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.
9. R. Laroia, *Structured Fixed-Rate Vector Quantizers Derived from Variable-Length Encoded Scalar Quantizers*, Ph.D Dissertation, Electrical Engineering Department, Univ. of Maryland, College Park, in preparation.

	Codebook Search	Encoding Operation
Computational Requirement (floating point operations)	$n(L+2)$ $\approx 1.5mbrn$ or $\approx$ Const.$\times mbr$ (for large r)	$n \times \frac{mr}{32}$, or $\approx$ Const.$\times mr$ (for large r)
Storage Requirement (32 bit numbers)	$L(1+\frac{m\log_2 n}{32})+n$ $\approx 1.5mbr(1+\frac{m\log_2 n}{32})+n$	$mL \times \frac{mr}{32}$ $\approx 1.5m^2br \times \frac{mr}{32}$

Table 1

Computational and Storage Requirments of the SVQ.

Rate	Block-Size					LMQ	ECSQ
	4	8	16	24	32		
$\alpha = 2.0$							
1.0	4.67	4.74	4.75	4.75	4.76	4.40	4.64
1.5	7.12	7.33	7.47	7.55	7.58	–	7.55
2.0	9.71	9.96	10.26	10.37	10.43	9.30	10.55
2.5	12.44	12.61	13.00	13.07	13.21	–	13.54
3.0	15.14	15.35	15.71	15.85	16.00	14.62	16.56
$\alpha = 1.0$							
1.0	4.80	5.23	5.47	5.60	5.61	3.01	5.76
1.5	6.59	7.51	8.03	8.14	8.22	–	8.55
2.0	9.23	9.75	10.42	10.57	10.73	7.53	11.31
2.5	11.49	12.26	12.98	13.21	13.31	–	14.32
3.0	14.13	14.95	15.66	15.91	16.05	12.61	17.20
$\alpha = 0.5$							
1.0	5.41	6.93	7.31	7.46	7.79	1.82	8.53
1.5	7.97	9.24	10.00	10.54	10.75	–	11.57
2.0	9.94	11.69	12.44	13.10	13.40	5.53	14.53
2.5	11.87	14.09	14.98	15.38	15.97	–	17.49
3.0	14.44	16.23	17.19	17.83	18.18	10.21	20.41

Table 2

SNR (in dB) of SVQ's for Three Different Generalized Gaussian Sources at Different Bit Rates (in bits/sample).

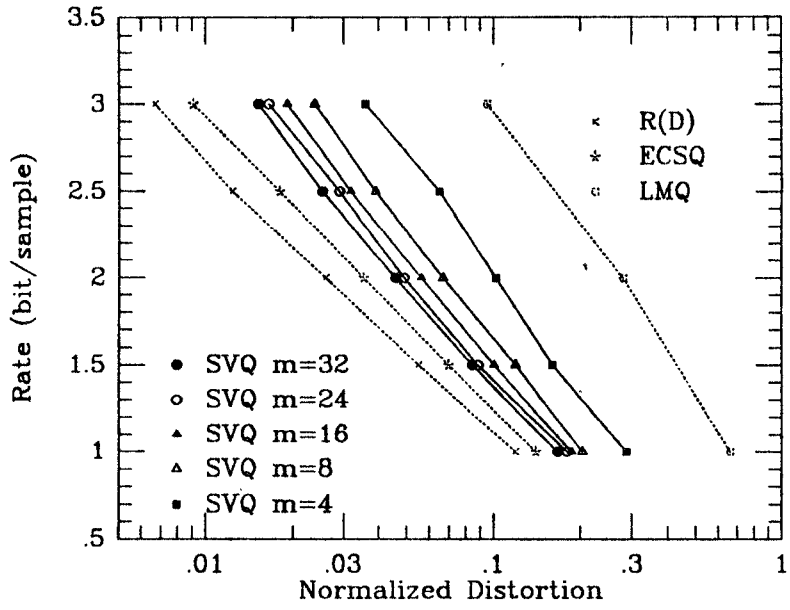

Fig. 1.a: Rate-Distortion Characteristics of the SVQ for a Generalized Gaussian Source; $\alpha = 0.5$.

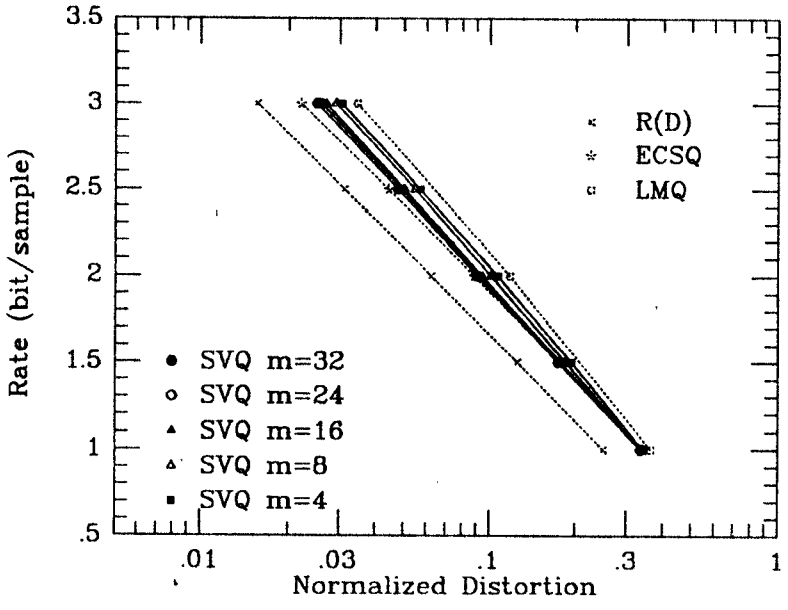

Fig. 1.b: Rate-Distortion Characteristics of the SVQ for a Generalized Gaussian Source; $\alpha = 2.0$.

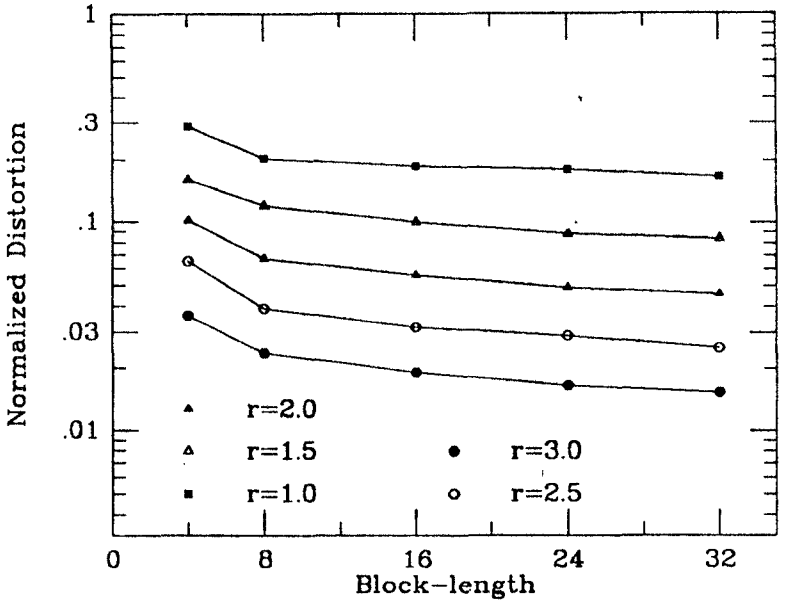

Fig. 2.a: Normalized Distortion of the SVQ as a Function of Block-length for a Generalized Gaussian Source; $\alpha = 0.5$.

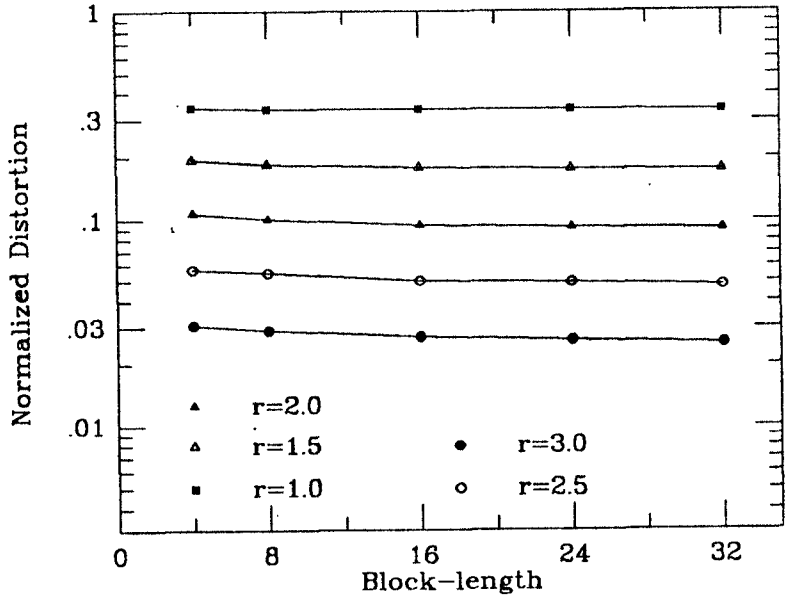

Fig. 2.b: Normalized Distortion of the SVQ as a Function of Block-length for a Generalized Gaussian Source; $\alpha = 2.0$.

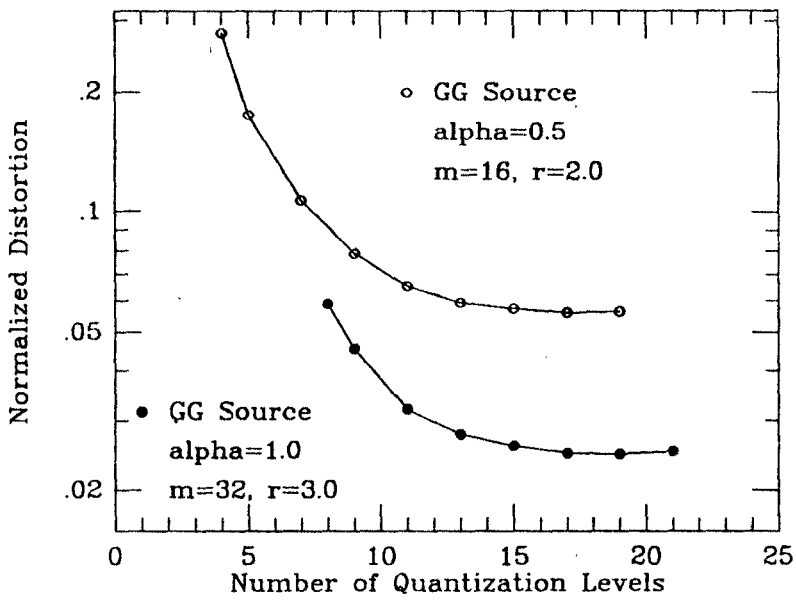

Fig 3: Normalized Distortion of the SVQ as a Function of the Number of Quantization Levels.

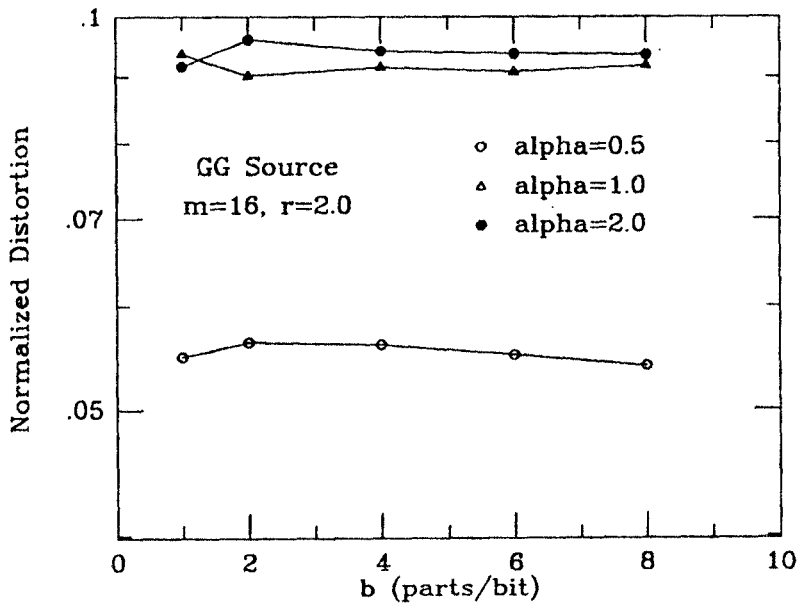

Fig 4: Variation of the SVQ Performance with Length Quantization.